

EIM Services
Thor Design Panel 2/3

November 18, 1997

Version 2.1

EIM and Application Interface Thread

Table of Contents

1. EIM Services	1
1.1 EIM Services Introduction.....	1
1.1.1 EIM Services Overview	1
1.1.2 EIM Services Operational Description.....	1
1.2 EIM Services Specifications	2
1.2.1 EIM Services Groundrules	2
1.2.2 EIM Services Functional Requirements	2
1.2.3 EIM Services Performance Requirements	8
1.2.4 EIM Services Interfaces Data Flow Diagrams.....	8
1.3 EIM Services Design Specification.....	9
1.3.1 EIM Services Detailed Data Flow.....	9
1.3.2 EIM Services External Interfaces	10
1.3.3 EIM Services Test Plan	16
1.3.4 Issues:	16

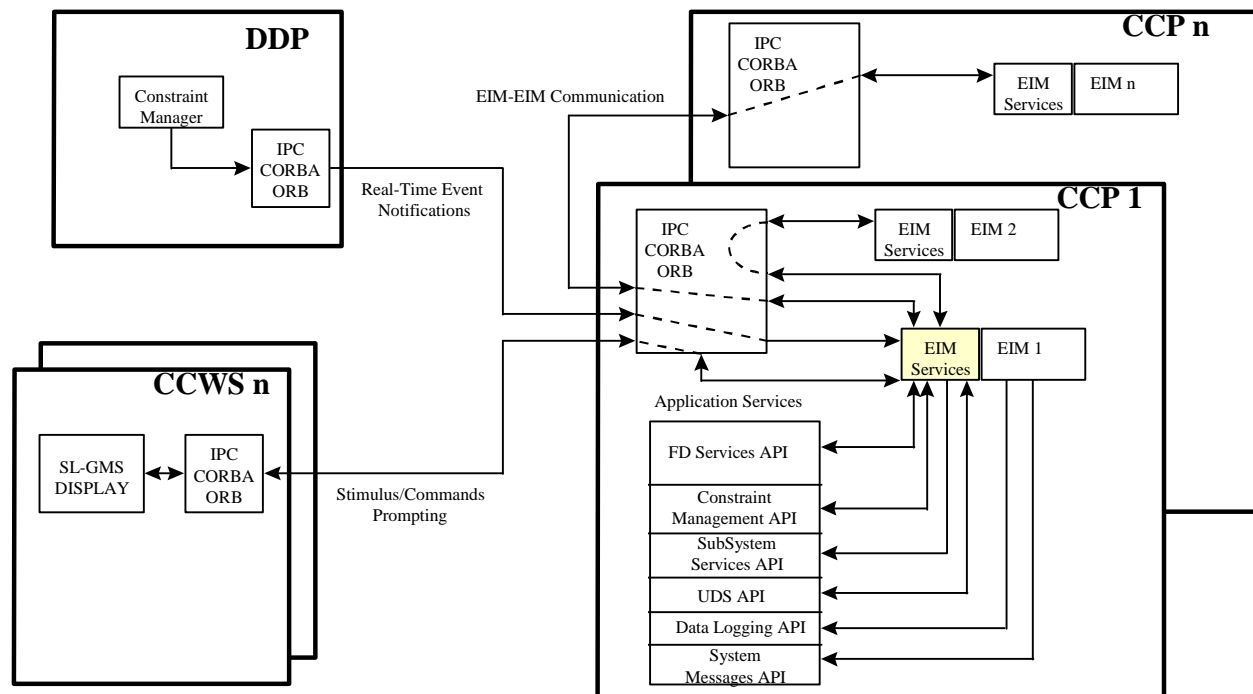
1. EIM Services

1.1 EIM Services Introduction

1.1.1 EIM Services Overview

The EIM Services CSC provides wrappers for EIMs to communicate with other Application Services on the RTPS. The wrappers provide an interface to FD Services, constraint management, user prompting and other systems and services required by the EIMs. EIM Services reside on the CCP within the scope of the EIM in ControlShell application development CASE tool. (EIM Services' components will be linked into the EIM application – not a separate process.) EIM Services provide ControlShell components which bring the Application Services into the visual framework of ControlShell. This allows the EIM designers to concentrate on the systems design and not the application services details. A repository of components will provide the required RTPS functions needed by EIMs.

End Item Management Services is a collection of components that encapsulate Application Service's APIs providing an interface between User EIM Applications and Application Services. This protects applications from any changes in the underlying Application Services interface, as well as protecting application services from changes in user application tools.



1.1.2 EIM Services Operational Description

The EIM Services CSC shall provide components for the EIM to communicate with Application Services on the CCP. Commands originating in the CCWS will interface with the EIM through CORBA. The EIM will provide control and monitoring with Finite State Machines (FSMs) and Composite Object Groups (COGs). COGs and FSMs will contain EIM Service's components for interfacing with Application Services. EIM Services shall call the API's to update FD values, send FD commands, set Constraints, display dialog windows. EIM Services shall use CORBA to receive and send stimuli/commands from other applications, receive Constraint Notifications, and for other inter-process communication.

1.2 EIM Services Specifications

1.2.1 EIM Services Groundrules

- Not all Applications Services API's or FD types will be provided in Thor
- The components are only a thin wrapper interface and will simply pass on any Application Service's return codes, errors back to the calling End Item Manager application.
- EIM Services makes no distinction between pseudo FD's, Fused FD's and real FD's.
- EIM Services will provide common components as needed by Application Development. For example:
 - Data logging components
 - System Message components
- EIM Services expects to be provided:
 - FD Services: API
 - EIM App Software: Requirements, Priorities
 - Constraint Management Services API and User Guide
 - ControlShell: Licenses, Training, Beta Release
- The End Item Manager test applications will be implemented by EIM Services.

1.2.2 EIM Services Functional Requirements

This section contains a list of SLS and high level derived requirements that are driving the design.

- (SLS 2.2.3.3.3) The CLCS shall provide the capability to restrict issuance of commands to only authorized users and applications.
- (SLS 2.2.4.1.1) The CLCS shall provide, using a Graphical User Interface (GUI) paradigm, the capabilities identified in Appendix C with a Y in the column titled Implement.
- (SLS 2.2.5.1.1) The RTPS shall provide fault tolerant End-Item monitoring and control.
- (SLS 2.2.5.1.2) The RTPS shall provide End-Item monitoring and control definition that is directly understandable in form and content by End-Item engineering personnel.
- (SLS 2.2.5.1.3) The RTPS shall provide a layered technique for defining End-Item monitoring and control that supports reuse and reduction of time to implement.
- (SLS 2.2.5.1.5) The RTPS shall provide the capability to monitor all End-Item FD's current state against expected state during all operational End-Item phases.
- (SLS 2.2.5.1.6) The CLCS shall provide effective support to test director and test management personnel to clearly understand End-Item and test summary status.
- (SLS 2.2.5.1.7) The CLCS shall provide effective support to test director and test management personnel to efficiently and reliably control and track testing progress.

Constraint Management:

- (SLS 2.2.5.4.2) CLCS shall provide the capability for multiple (TBD number) users and system or user applications to request notification of constraint events for each Measurement FD.
- (SLS 2.2.5.4.3) CLCS shall provide the capability for each user, and system or user application requesting constraint notification to specify the limits/condition under which they will be notified.

End-Item Management:

The End-Item Management function provides the capability for users to create End-Item Manager applications to perform closed loop control of a specific End-Item system or component. A hierarchy of End-Item Managers might exist for large and complex End-Item Systems.

- (SLS 2.2.5.6.1) RTPS shall provide the capability to delegate and define closed loop control of one or more End-Items to an End-Item Manager.
- (SLS 2.2.5.6.2) The RTPS End-Item Manager function shall provide the capability to automatically safe the End-Item and continue if commanded when a RTPS system fault occurs.
- (SLS 2.2.5.6.3) RTPS shall provide the capability to delegate continuous End-Item monitoring to Constraint Management and respond to Constraint Management notification events.
- (SLS 2.2.5.6.4) RTPS shall provide the capability to operate an End-Item using rate based control.
- (SLS 2.2.5.6.5) RTPS shall provide the capability for an End-Item Manager Test Application to issue a command to another End-Item Manager Test Application.
- (SLS 2.2.5.6.6) RTPS shall provide the capability to alter the state of frame rate/time domain control based on an event.
- (SLS 2.2.5.6.7) RTPS shall provide priority Reactive Sequence control processing for End-Item Managers requiring minimum reaction time to Constraint Management notification events.
- (SLS 2.2.5.6.8) RTPS shall provide an End-Item control layer/encapsulation allowing maintenance in a single place and reuse through out user applications.

This section is derived requirements taken from the ASV API matrix. The requirement numbers are from the User Apps Document (UAD) Requirement Number Column.

- (UAD 4.2.1.1.1) *The API shall provide a method to issue values to analog output FD's to support on-board port/MDM specifications.*
- (UAD 4.2.1.5) *The capability shall be provided to support on-board/MDM specifications in discrete commands.*
- (UAD 4.2.4.1.11) *The API shall provide a method for determining which system asserted a constraint when an FD is marked with a constraint limit violation.*
- (UAD 4.2.4.1.x) *The API shall provide a method for changing the constraint condition for an enumerated FD.*
- (UAD 4.2.4.3.2) *The API shall provide a method for changing the significant change value of an analog FD.*
- (UAD 4.2.4.3.3) *The API shall provide a method for changing the stale data count for any FD.*
- (UAD 4.2.4.3.4) *The API shall provide a method for changing calibration coefficients.*
- (UAD 4.2.4.3.7) *The API shall provide a method to activate or inhibit stale data checking on a per FD basis.*
- (UAD 4.2.4.3.9) *The API shall provide a method for reading the current significant change value for an analog FD.*
- (UAD 4.2.4.3.10) *The API shall provide a method to read the current stale data count for any FD.*
- (UAD 4.3.1.4) *The API shall provide a method to activate or inhibit stale data checking on any gateway.*
- (UAD 4.3.1.5) *The API shall provide a method to activate or inhibit data fusion limit checking for any class of limits.*
- (UAD 4.10.19) *Functionality shall be provided to control the SRB MDM's (lock/unlock). (Reference KSC-LPS-OP-033-4 Section 6.0)*
- (UAD 4.2.2.2) *The API shall provide a method for reading the current value of an analog FD in raw counts.*

- (UAD 4.2.2.5) *The API shall provide a method for reading the current value of a discrete FD in unprocessed format.*
- (UAD 4.7.2.2) The API shall provide a method to set the systems CDT/MET.
- (UAD 4.7.2.5) The API shall provide a method for reading the system CDT/MET.
- (UAD 4.7.2.6) The API shall provide a method to start/hold the system CDT/MET.
- (UAD 4.7.2.5) The API shall provide a method to specify and cancel event notification and an event handler for the arrival of a specific CDT or MET.
- (UAD 4.2.1.7) The API shall support one-to-one and one-to-many options for the issuance of values to FD's
- (UAD 4.2.2.4) The API shall provide a method for reading the current value of a discrete FD in engineering units.
- (UAD 4.2.2.6) The API shall provide a method to read the time of the last change in value of the FD.
- (UAD 4.2.1.6) The API shall provide a method for issuing a value to digital pattern output FD's.
- (UAD 4.2.1.8) The API shall validate that issued values are compatible with Function Designator types prior to issuing the command.
- (UAD 4.1.1.2) The API shall be able to pass engineering unit data types across interface boundaries in a type safe way.
- (UAD 4.1.2) The API shall provide an efficient type safe mechanism for manipulating computations involving engineering units.
- (UAD 4.2.1.2) The API shall provide a method for setting a discrete output FD's.
- (UAD 4.2.2.1) The API shall provide a method for reading the current value of an analog FD in engineering units.
- (UAD 4.2.2.7) The API shall provide a method to read the health status the last change of an FD.
- (UAD 4.2.3.1) "The API shall provide a method to identify function designators that shall be delivered via queued service, providing access to every change value in time ordered fashion"
- (UAD 4.2.3.2) The API shall provide a method to read the next value of a multi-sample queued function designators.
- (UAD 4.2.3.3) The API shall provide a method to read the next N values of a multi-sample queued function designator.
- (UAD 4.2.3.4) The API shall provide a method to clear all queued samples pending for the application.
- (UAD 4.2.3.5) The API shall provide a method to cancel queued function designator delivery by FD.
- (UAD 4.2.5.1) The API shall provide a method for querying the on-line data bank by FDID or FD name.
- (UAD 4.2.5.2) The API shall provide a method for querying any piece of information stored in the on-line data bank for a particular FD.
- (UAD 4.4.1.5) "The API shall provide a method to issue a prompt to, and receive response from, a user display window."
- (UAD 4.1.1.1) "The API shall preserve the current LPS engineering unit model. The API shall be able to work with temperatures, pressures, discrete states, and enumerated states."
- (UAD 4.6.2) The API shall provide a method to translate API error codes into system text messages which may be displayed for the user.
- (UAD ???) The API shall provide a method for reading the return to limits indicator (event) of an digital pattern / Enumerated FD.

- (UAD 4.2.4.1.10) The API shall provide a method for reading an FD's high limit constraint indicator (event) for all analog constraint limit sets.
- (UAD 4.2.1.1) The API shall provide a method to apply values to analog output FD's.
- (UAD 4.2.1.3) "The API shall set discrete output FD's using the literal key words OPEN, CLOSE, TRUE, FALSE, WET, DRY, ON, OFF."
- (UAD 4.2.1.4) The API shall provide a method to specify a time value for a discrete command. This command shall set the command to the indicated state for the specified period and then return it to the original state.
- (UAD 4.2.1.9) The API shall provide a method for issuing values to a pseudo function designator.
- (UAD 4.2.4.1.1) The API shall provide a method for changing the constraint limits associated with an analog FD.
- (UAD 4.2.4.1.2) The API shall provide a method for changing the constraint state of a discrete FD.
- (UAD 4.2.4.1.3) The API shall provide a method for changing the constraint condition for a digital pattern FD.
- (UAD 4.2.4.1.4) The API shall provide a method to activate or inhibit constraint checking associated with an FD for an application.
- (UAD 4.2.4.1.7) The API shall provide a method for reading the FD constraint indicator (event) for an FD.
- (UAD 4.2.4.1.8) The API shall provide a method for reading the return to limits indicator (event) of an FD.
- (UAD 4.2.4.1.9) The API shall provide a method for reading an FD's low limit indicator (event) for all analog constraint limit sets.
- (UAD 4.5.1.4) The API shall provide methods for communicating between concurrently executing applications as described in 4.5.1.3 above.
- (UAD 4.2.2.3) The API shall provide a method for reading the current value of a digital pattern FD and enumerated type FD.
- (UAD 4.2.4.4.1) "The API shall provide a method for determining if a function designator's health status is OK, FAILED, or WARNING."
- (UAD 4.2.4.4.2) The API shall provide a method of reading the detailed health status from the health status word for a function designator. The detailed health shall identify the following conditions:
- (UAD 4.2.4.4.2.2) Is processing active or inhibited for this FD?
- (UAD 4.2.4.4.2.6) Is application advisory notification active or inhibited for this FD?
- (UAD 4.2.4.4.2.7) Is engineering bypass active or inhibited for this FD?
- (UAD 4.2.4.4.4) The API shall provide a method to change an FD's status indicator (event) in order to mark the measurement bad or good.
- (UAD 4.5.2.4) The API shall provide a method to specify and cancel event notification and an event handler for the occurrence of an FD constraint violation.
- (UAD 4.5.2.8) The API shall provide a method to activate or inhibit all FD constraint notifications active for the application.
- (UAD 4.1.4) The API call shall return an indicator to the calling process which indicates success or failure of an API call and the reason associated with any failure condition.
- (UAD 4.1.5) The API shall provide a method of indicating why a command has failed. This status shall indicate the following error conditions:
- (UAD 4.2.4.1.5) The API shall provide a method for reading the constraint limits associated with an analog FD for an application.
- (UAD 4.2.4.3.1) The API shall provide a method for changing the sample rate of any GSE FD.

- (UAD 4.2.4.1.x) The API shall provide a method for reading the constraint conditions associated with a discrete FD for an application.
- (UAD 4.2.4.3.8) The API shall provide a method of reading the current sample rate of an FD.
- (UAD 4.3.2.5) The API shall provide a method to read the constraint limit processing status in the data fusion function.
- (UAD 4.1.3) The API shall support strong compile time checking and external symbol resolution. The API shall minimize the need for run-time dependency checking.
- (UAD 4.2.4.3.6) The API shall provide a method to activate or inhibit measurement processing on a per FD basis.
- (UAD 4.2.4.3.11) The API shall provide the capability to read the current hardware address of an FD.
- (UAD 4.2.4.4.3) The API shall provide a method for reading the current data stale indicator for an FD.
- (UAD 4.6.3) The API shall provide a method to specify application error routines which are called on the occurrence of user specified error conditions.

This section contains derived requirements not captured in the SVA API matrix:

1. EIM Services will provide ControlShell repositories:
 - A development repository
 - A user repository (Configuration controlled).
2. EIM Services will provide timing functionality:
 - Stopwatch / elapsed time
 - Timeout

FD Services:

3. EIM Services will provide a ControlShell custom type for each FD type: (Thor minimum)
 - Discrete
 - ON_OFF
 - OPEN_CLOSED
 - TRUE_FALSE
 - WET_DRY

- *Analog*s

AMP	AMPAC	<i>ARC_MIN</i>	<i>ARCS</i>
<i>AU</i>	<i>BFS</i>	<i>C</i>	<i>CNT</i>
<i>DB</i>	DEG	<i>DEG_HR</i>	<i>DEG_HR_G</i>
<i>DEG_S</i>	<i>DEGC</i>	DEGF	<i>DEGK</i>
<i>DEGR</i>	<i>FT</i>	<i>FT_S</i>	<i>FT_S2</i>
<i>G</i>	<i>GAL</i>	GAL_MIN	<i>GHZ</i>
<i>GM</i>	<i>GMT</i>	<i>GP_P</i>	<i>GR_LBDA</i>
<i>HZ</i>	<i>IN</i>	<i>INH20</i>	<i>INHG</i>
<i>KFT</i>	<i>KFT_S</i>	<i>KGAL</i>	<i>KGAL_MIN</i>
<i>KHZ</i>	<i>KLBF</i>	<i>KT</i>	<i>KT_S</i>
<i>KUG</i>	<i>KUG_G</i>	<i>KW</i>	<i>KW_M2</i>
<i>LBF</i>	<i>LBM</i>	<i>LBM_F2_S</i>	<i>LBM_FT2</i>
<i>LBM_FT3</i>	<i>LBM_HR</i>	<i>M</i>	<i>M_S</i>
<i>MACH</i>	MAMP	<i>MEGAFT</i>	<i>MEGAFT_S</i>
<i>MEGAHZ</i>	<i>MI</i>	<i>MI_S</i>	<i>MJ</i>
<i>MM</i>	<i>MMHG</i>	<i>MRAD</i>	<i>MRAD_S</i>
<i>MV</i>	<i>NM</i>	<i>OHM</i>	<i>P</i>
PCT	<i>PPM</i>	<i>PPS</i>	<i>PSI</i>
<i>PSI_MIN</i>	PSIA	PSID	PSIG
<i>RAD</i>	<i>RAD_S</i>	<i>RAD_S_G</i>	<i>RAD_S2</i>
<i>RPM</i>	<i>SLUGS</i>	<i>TORR</i>	<i>UG</i>
<i>UG_G</i>	<i>UGM_3</i>	<i>UIN</i>	<i>UIN_IN</i>
<i>UM</i>	<i>UM_M</i>	<i>UNITS</i>	V
VAC	<i>VP_P</i>	<i>VRMS</i>	<i>W</i>

- *Digital Patterns*

BIN
OCT
DEC
HEX

4. EIM Services will for each FD type provide an ATC and custom interface for FD Services Interface Agreement (KSC-84K00362):
 - Read Measurement FDs
 - Write Stimulus FDs
 - Read Multi Sample Queued FDs
 - Read static FD information
5. ATCs shall provide access to selected FD Object methods and data. The interface will be implemented with ControlShell components in a centralized repository directory.

Constraint Management:

6. EIM Services shall provide a capability for an EIM to Assert, Alter, Acknowledge, and Release Constraint Checking for an FD.
7. Supports Multiple Constraints per FD, but each calling component must maintain the Constraint ID(s) related to its constraint(s)
8. Constraint Interface will be contained in FD ATCs
9. EIM Services shall provide a capability to receive Real-time Constraint Notifications via CORBA interface from Constraint Management.

10. Notifications shall be distributed with a ControlShell Import Bubble on the interface

11. Inter-Process Communications:

12. EIM Services shall provide ControlShell components for EIM construction, allowing EIMs to interact with other processes, implemented via CORBA.

13. User Display Services:

14. EIM services shall provide ControlShell component wrappers to the Application service's User Display Services API's:

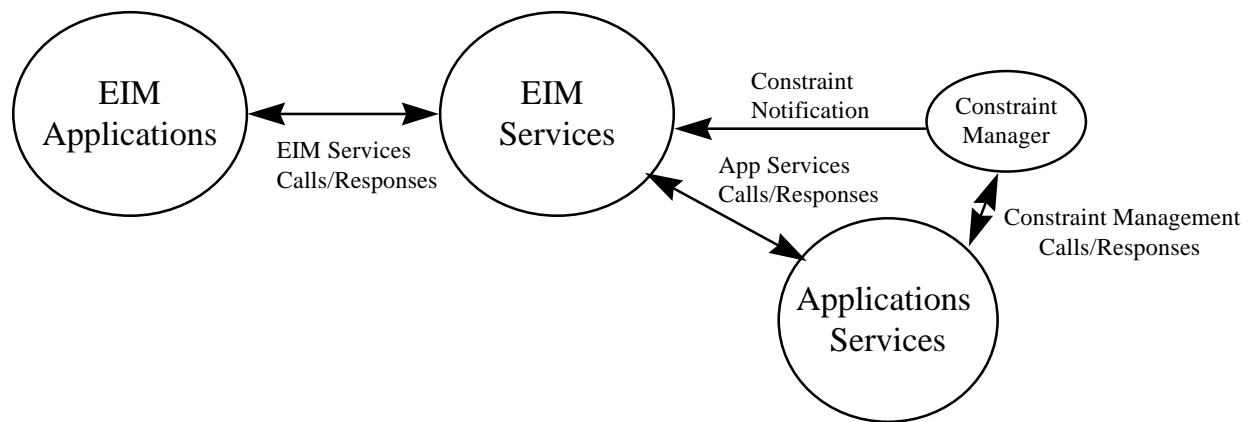
- Multi-button prompt Dialog
- *Input Dialog*
- *Message Dialog*

1.2.3 EIM Services Performance Requirements

None identified for Thor.

1.2.4 EIM Services Interfaces Data Flow Diagrams

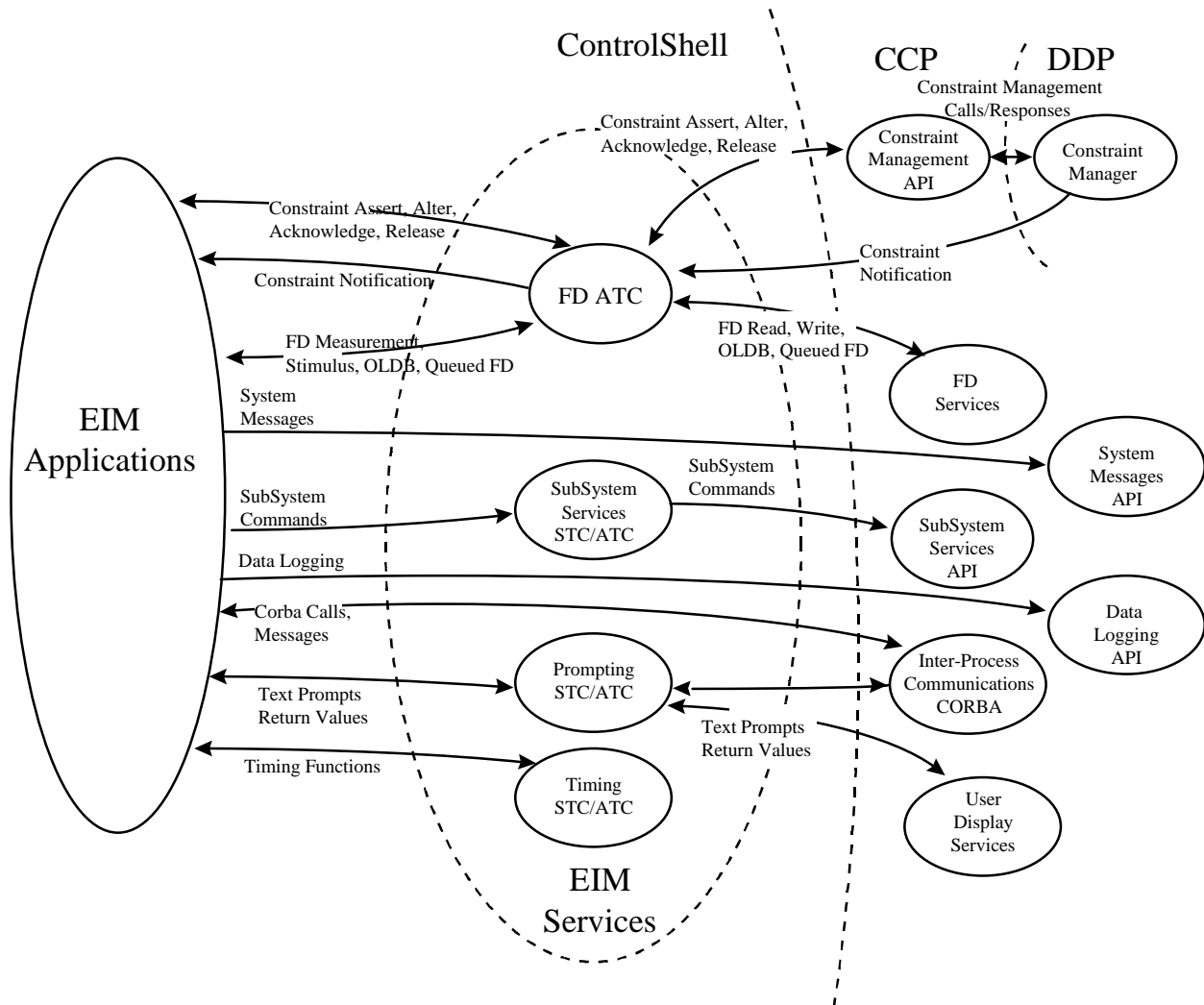
This diagram provides a pictorial representation of the data flow between EIM Services, EIM Applications and Application Services.



1.3 EIM Services Design Specification

1.3.1 EIM Services Detailed Data Flow

This diagram provides a pictorial representation of the data flow between EIM Services, EIM Applications and Application Services objects.

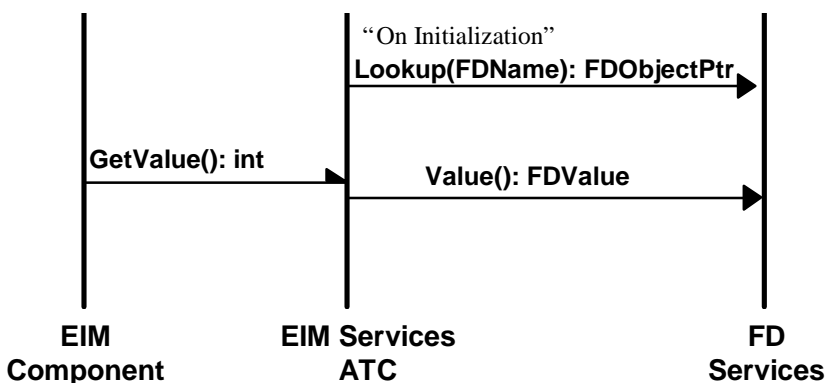


1.3.2 EIM Services External Interfaces

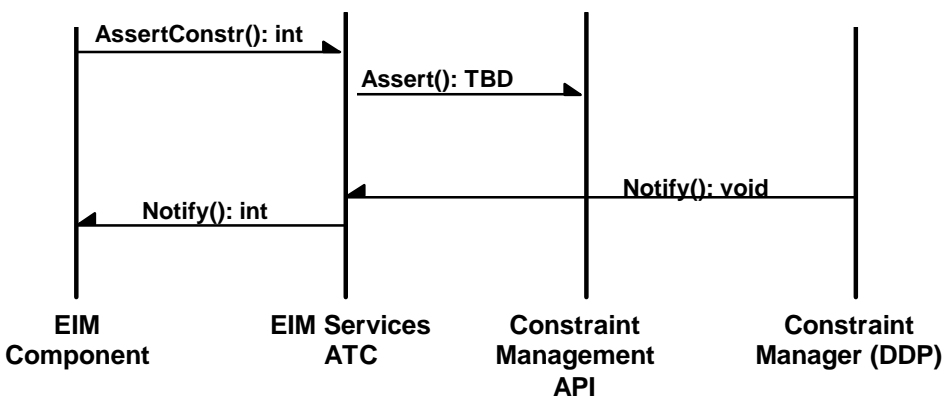
Operational Scenarios:

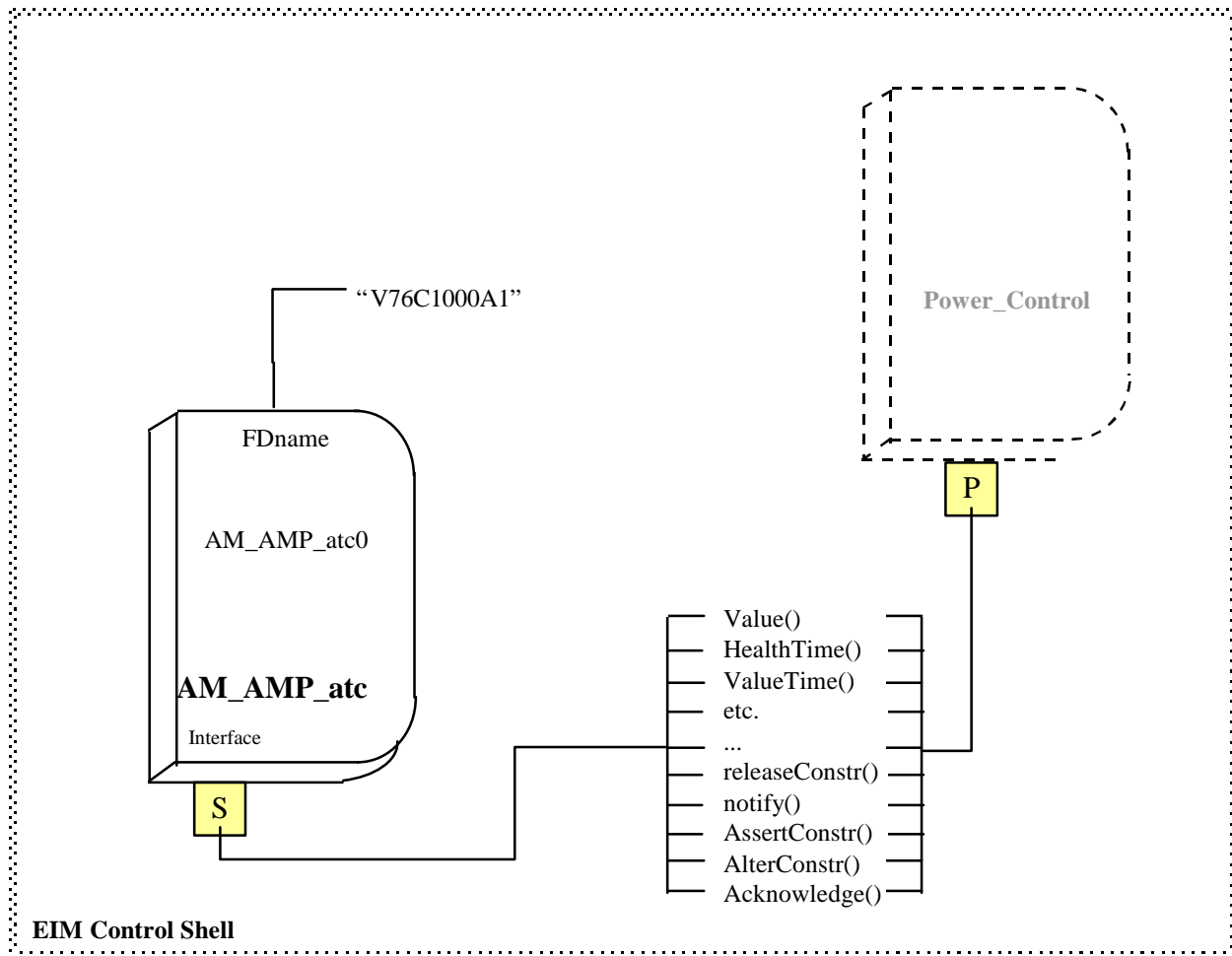
During a typical development of an End Item Manager application utilizing FD services, an atomic component is selected from the repository and placed in the ControlShell graphical environment where a call to an FD Service is needed. The specific ATC for the needed FD service and FD type involved must be correctly selected, i.e. command capability of an analog AMP type. The FD name string must be wired to the ATC's FDname reference pin. Also ATC's interface must be wired to the calling component.

During execution, the calling component will activate any of the ATC's methods which will perform the associated FD services API call and return the results.



The EIM Application shall perform all the required Constraint functions through the Constraint Management (CM) Services Interface on each FD ATC. Putting the Constraint functions on each FD will allow the constraint functions to be statically tied to their respective FD. Constraints shall be applied, queried, altered, and released through the methods encapsulated as bubbles within the Constraint Interface. When a Constraint is Asserted in an EIM component, the (EIM Services) FD Services ATC calls the CM API method `Assert()`. Then Constraint Manager will set the constraint and a CM Event Notification verifying the assertion will be received in the FD ATCs through a CORBA interface. The ATC will then call the `Notify` method on all components connected to its interface completing the `Assert` method. When a constraint event occurs, the Constraint Manager will send a CM Event Notification through CORBA to the ATC and the ATC will send the message to the EIM components with the `Notify` method, as before. The EIM Application must provide the `Notify` method coding to handle the notification function call. The EIM Application will receive the entire CM Event Notification including all CM provided fields and it will be up to the EIM Application to keep track of the Constraint Ids of each of the Constraints it sets.





Example Code:

```
#include "asv_fdsFunctionDesignator.h"
#include "asv_fdsFDDictionary.h"
#include "cs.h"                                // ControlShell header

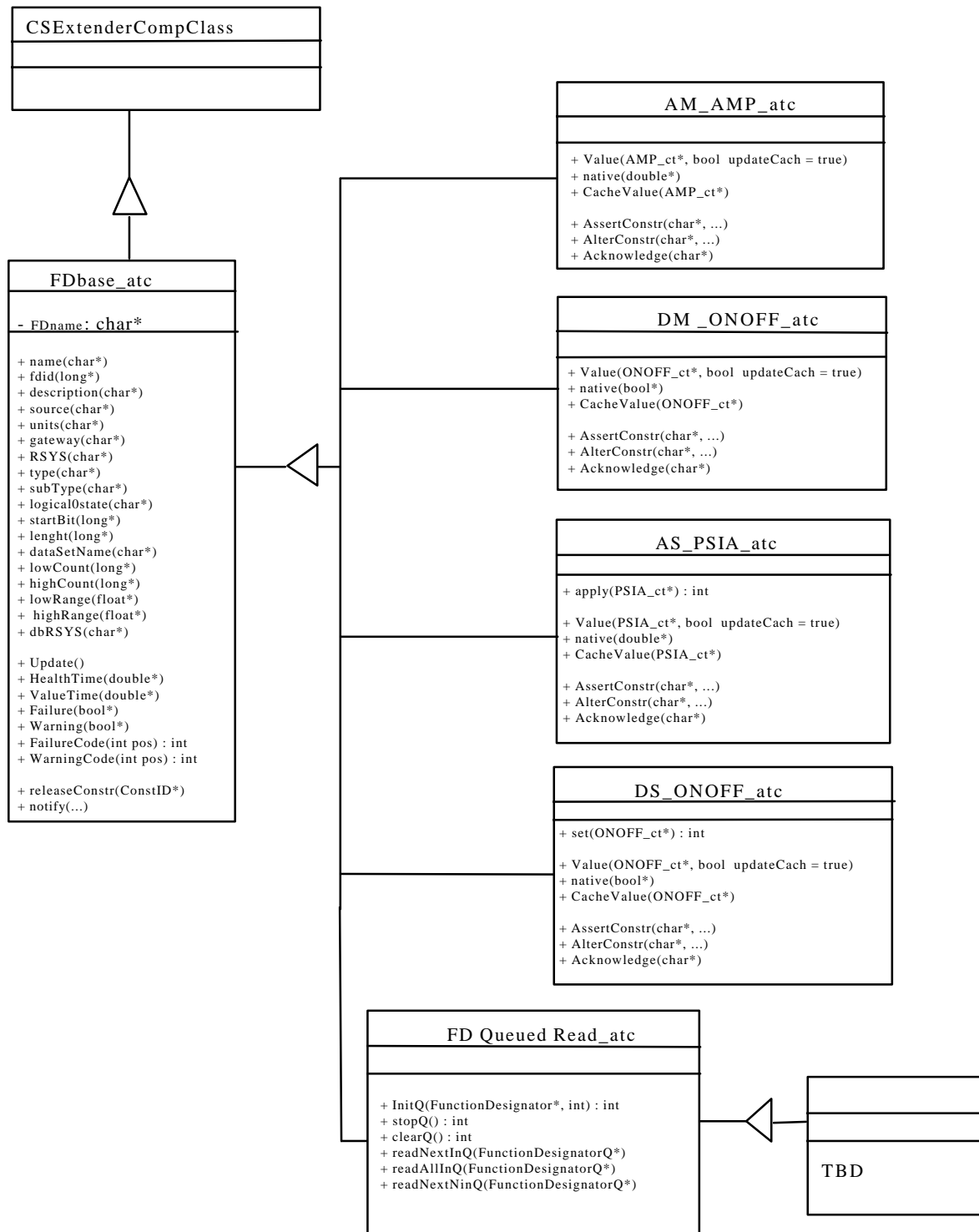
extern FDDictionary d;

void AM_AMP_atc::onInstance()
{
    FDObjPtr = d.lookup( FDname ); //get ptr to FD object by FD lookup by name
}

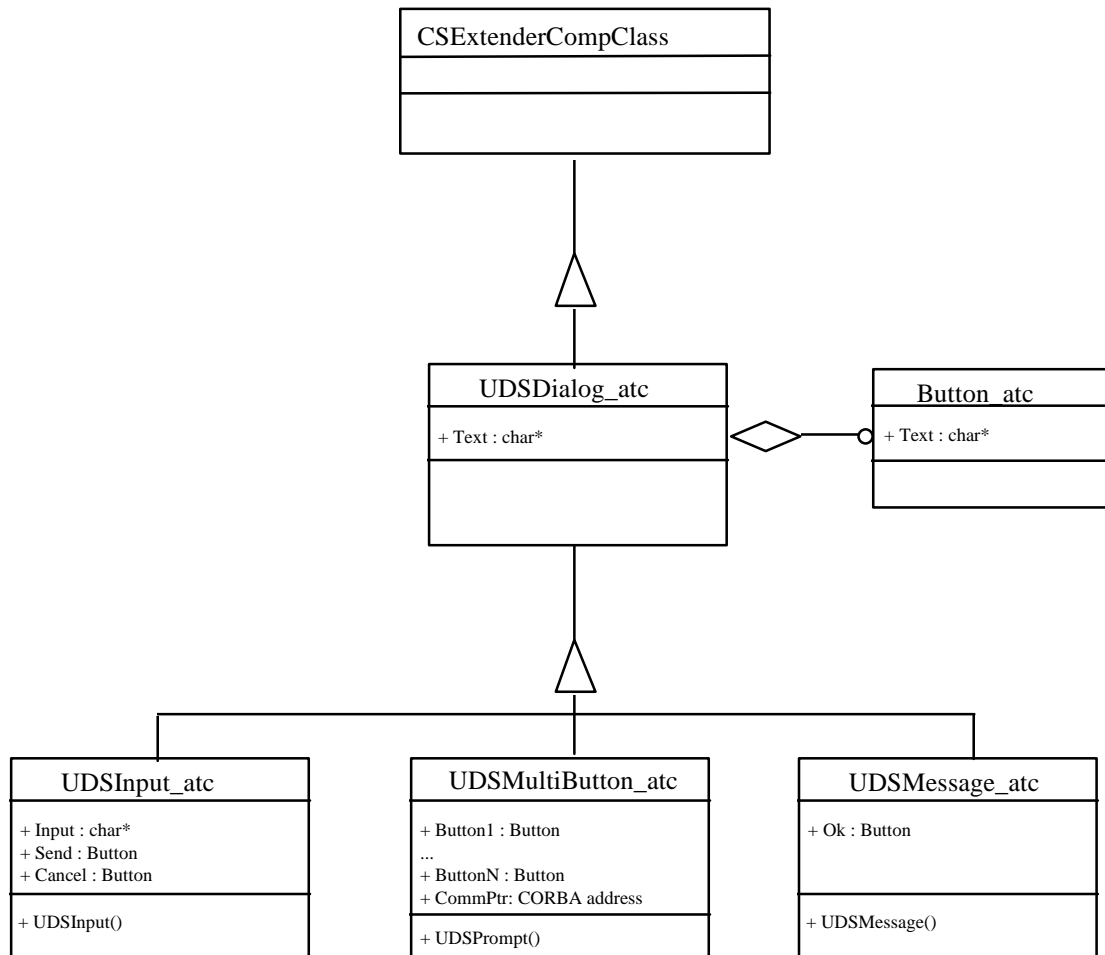
void AM_AMP_atc::Value(AMP *amp_ptr)
{
    amp_ptr->_val = FDObjPtr ->Value(); // FD Services API call
    return ;
}
```

1.3.2.1 EIM Services Class Hierarchy

ControlShell FD Services Wrappers (ATCs) - Class Hierarchy Diagram



ControlShell User Display Dialog Services Wrappers (ATCs) - Class Hierarchy Diagram



1.3.2.2 EIM Services Display Formats

N/A

1.3.2.3 EIM Services Input Formats

N/A

1.3.2.4 Recorded Data

N/A

1.3.2.5 EIM Services Printer Format

N/A

1.3.2.6 Interprocess Communications (C-to-C Communications)

All inter-process communication is via CORBA.

1.3.2.7 EIM Services External Interface Calls (e.g., API Calling Formats)

This is the list of methods available to a calling EIM component.

FD Measurement ATC interfaces for FD AMP type:

Value(AMP_ct*, bool updateCach = true) - method to call FD services and return FD value
native(double*) - method to call FD services and return FD value in native C++ type
CacheValue(AMP_ct*) - method to call FD services and return FD value from cache

Update() - method to call FD services and perform atomic read of FD data from CVT.
HealthTime(double*) - method to call FD services and return health update time.
ValueTime(double*) - method to call FD services and return value update time.
Failure(bool*) - method to call FD services and return failure health bit.
Warning(bool*) - method to call FD services and return warning health bit.
FailureCode(int pos) : int - method to call FD services and return failure code.
WarningCode(int pos) : int - method to call FD services and return warning code.

name(char*) - method to call FD services and return FD name.
fdid(long*) - method to call FD services and return real time control net ID.
description(char*) - method to call FD services and return FD description.
source(char*) - method to call FD services and return source of FD data.
units(char*) - method to call FD services and return engineering units of FD.
gateway(char*) - method to call FD services and return gateway processor ID.
RSYS(char*) - method to call FD services and return TCID responsible system.
type(char*) - method to call FD services and return FD data type.
subType(char*) - method to call FD services and return FD data subtype.
logical0state(char*) - method to call FD services and return logical zero state.
startBit(long*) - method to call FD services and return starting bit position of FD data.
length(long*) - method to call FD services and return FD data length.
dataSetName(char*) - method to call FD services and return time homogenous data set name.
lowCount(long*) - method to call FD services and return low count limit.
highCount(long*) - method to call FD services and return high count limit.
lowRange(float*) - method to call FD services and return low engineering value limit.
highRange(float*) - method to call FD services and return high engineering value limit.
dbRSYS(char*) - method to call FD services and return Databank responsible system.

ReleaseConstr(const ConstraintID IN, const EventID IN)

Notify(ConstraintID, Time, State, Value)

AcknowledgeConstr(const ConstraintID IN, ConstraintExpression)

AssertConstr(Algorithm, HealthCheck, BoundaryCheck, NotificationFlag, Category, AcknowledgeReq, InputArgs, EventID);

RETURNED IN NOTIFICATION OBJECT:

Status, ErrorCode, ConstraintID, EventID

AlterConstr(ConstraintID, AlterOption, AlterFieldValues, EventID)

RETURNED IN NOTIFICATION OBJECT:

Status, ErrorCode, ConstraintID, EventID

FD Stimulus ATC interface for FD type PSIA:

apply(PSIA_ct*) : int - method to call FD services and command an FD

 Value(PSIA_ct*, bool updateCach = true) - method to call FD services and return FD value
 native(double*) - method to call FD services and return FD value in native C++ type
 CacheValue(PSIA_ct*) - method to call FD services and return FD value from cache

 Update() - method to call FD services and perform atomic read of FD data from CVT.
 HealthTime(double*) - method to call FD services and return health update time.
 ValueTime(double*) - method to call FD services and return value update time.
 Failure(bool*) - method to call FD services and return failure health bit.
 Warning(bool*) - method to call FD services and return warning health bit.
 FailureCode(int pos) : int - method to call FD services and return failure code.
 WarningCode(int pos) : int - method to call FD services and return warning code.

 name(char*)
 fdid(long*)
 description(char*)
 source(char*)
 units(char*)
 gateway(char*)
 RSYS(char*)
 type(char*)
 subType(char*)
 logical0state(char*)
 startBit(long*)
 lenght(long*)
 dataSetName(char*)
 lowCount(long*)
 highCount(long*)
 lowRange(float*)
 highRange(float*)
 dbRSYS(char*)

 ReleaseConstr(const ConstraintID IN, const EventID IN)
 Notify(ConstraintID, Time, State, Value)
 AcknowledgeConstr(const ConstraintID IN, ContraintExpression)

 AssertConstr(Algorithm, HealthCheck, BoundaryCheck, NotificationFlag, Category, AcknowledgeReq, InputArgs,
 EventID);
 RETURNED IN NOTIFICATION OBJECT:
 Status, ErrorCode, ConstraintID, EventID

 AlterConstr(ConstraintID, AlterOption, AlterFieldValues, EventID)
 RETURNED IN NOTIFICATION OBJECT:
 Status, ErrorCode, ConstraintID, EventID

User Display Input Dialog ATC:

UDSInput() - method to call UDS and create an Input dialog box.

User Display Two Step Dialog ATC:

UDSMultiButton() - method to call UDS and create a multi-button dialog box.

User Display Message Dialog ATC:

UDSMessage()

- method to call UDS and create a message dialog box.

Timing Component:

elapsedTime()

- method to return elapsed time of an event (stopwatch).

timeOut()

- method to return a timer “interrupt.”

1.3.2.8 EIM Services Table Formats

N/A

1.3.3 EIM Services Test Plan

EIM Service’s components to FD Services system-level tests may be run in either or both the IDE or SDE environments. These tests are run on the basic CCWS, CCP and DDP platforms. There are no specific hardware configurations required. The minimal applications software configuration includes Data Distribution, Application Services and any programs and files necessary to run the CCP. EIM Service’s components to FD Services testing also requires a CLCS application or a EIM application test tool that exercises the FD read, FD write, OLDB read, and Queued Multi-Sample FD services. A test EIM Application will be required for the testing. At a minimum the test application shall have components of each type of FD and a test ControlShell FSM.

The specific test cases that will be run include:

1. read FD value and verify type safe data manipulations
2. command FD and verify that FD CMD is called.
3. read queued multi-sample FD data.
4. read OLDB provided data.
5. Assert a constraint on an FD, using the Constraint Viewer to verify
6. Repeat Test #1 and then Alter the Constraint
7. Assert two constraints on a single FD
8. Assert a constraint, force the value out of constraint, and then verify the receipt of the Constraint Notification
9. Release a Constraint
10. Using an FSM Assert several constraints and have the FSM respond to Constraint Notifications by changing states or performing some operation.
11. Turn data logging on & off and verify log.
12. Issue/Send system messages and verify receipt.
13. Generate non-FD command and verify receipt.
14. Inter-EIM commanding, verify second EIM receives (CORBA) command.
15. Inter-process communication, verify second process receives (CORBA) message.
16. Generate dialog windows and verify button actions.
17. All tests will include in-range and out of range values.

1.3.4 Issues:

1. The FD Queued Read functionality has not yet been defined, and barring resolution will not be provided for Thor.
2. Should User Display Services or custom ControlShell/CORBA be used for prompting display dialogs?
3. The repository directory structure has yet to be finalized.
4. Error and exception handling, processed in EIM Services components?
5. Timing capabilities required by EIMs yet to be finalized.
6. EIMs expect to be provided enumerated FD types.